PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06F

A2

(11) International Publication Number: WO 94/25913

(43) International Publication Date: 10 November 1994 (10.11.94)

(21) International Application Number: PCT/US94/04499

(22) International Filing Date: 29 April 1994 (29.04.94)

(30) Priority Data:
08/056,333 30 April 1993 (30.04.93) US

(71) Applicant: NOVADIGM, INC. [US/US]; Suite 200, One International Boulevard, Mahwah, NJ 07495 (US).

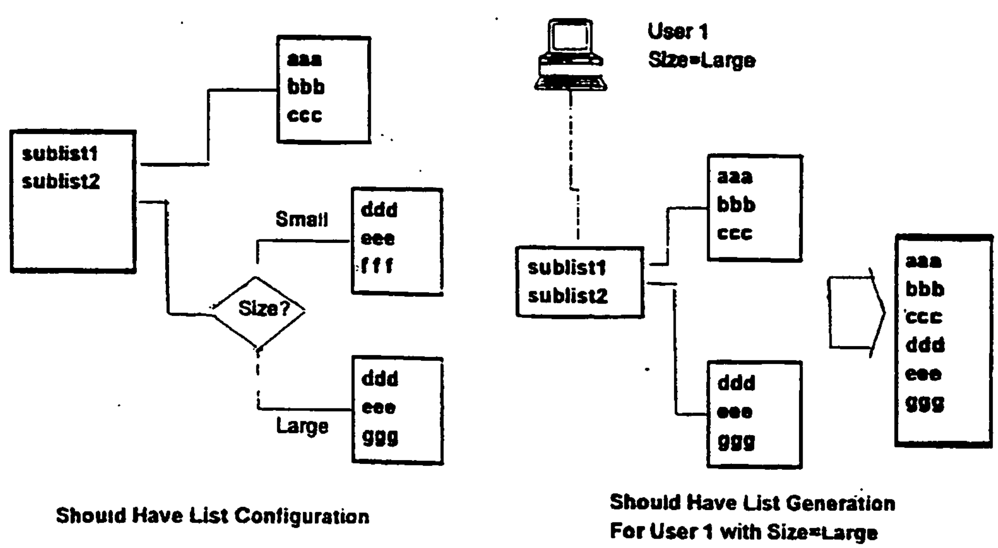
(72) Inventors: FITZGERALD, Albion, J.; 200 Melrose Place, Ridgewood, NJ 07450 (US). FITZGERALD, Joseph, J.; 24 Bonticou View Drive, New Paltz, NY 12561 (US).

(74) Agent: DURANT, Stephen, C.; Wilson, Sonsini, Goodrich & Rosati, 650 Page Mill Road, Palo Alto, CA 94304-1050 (US).

(81) Designated States: CA, JP, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: METHOD AND APPARATUS FOR ENTERPRISE DESKTOP MANAGEMENT



(57) Abstract

In an enterprise-wide network which includes at least one centralized computer and a plurality of desktop computers, a method for enterprise system management comprising the steps of: storing an Already Have list for each desktop; storing a plurality of Should Have sub-lists; and generating a respective Should Have list from the stored sub-lists for a respective desktop computer during configuration of the desktop computer; wherein the schema of the gnerated Should Have list includes at least one dynamic linkage which encompasses more than one Should Have sub-lists.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	GB	United Kingdom	MR	Mauritania
AU	Australia	GE	Georgia	MW	Malawi
BB	Barbados	GN	Guinea	NE	Niger
BE	Belgium	GR	Greece	NL	Netherlands
BF	Burkina Faso	HU	Hungary	NO	Norway
BG	Bulgaria	IE	Ireland	NZ	New Zealand
BJ	Benin	IT	Italy	PL	Poland
BR	Brazil	JP	Japan	PT	Portugal
BY	Belarus	KE	Kenya	RO	Romania
CA	Canada	KG	Kyrgystan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo	KR	Republic of Korea	SE	Sweden
CH	Switzerland	KZ	Kazakhstan	SI	Slovenia
CI	Côte d'Ivoire	LI	Liechtenstein	SK	Slovakia
CM	Cameroon	LK	Sri Lanka	SN	Senegal
CN	China	LU	Luxembourg	TD	Chad
CS	Czechoslovakia	LV	Latvia	TG	Togo
CZ	Czech Republic	MC	Monaco	TJ	Tajikistan
DE	Germany	MD	Republic of Moldova	TT	Trinidad and Tobago
DK	Denmark	MG	Madagascar	UA	Ukraine
ES	Spain	ML	Mali	US	United States of America
FI	Finland	MN	Mongolia	UZ	Uzbekistan
FR	France			VN	Viet Nam
GA	Gabon				

METHOD AND APPARATUS FOR ENTERPRISE DESKTOP MANAGEMENT

1. Field of the Invention

The invention relates generally to computer networks, and more particularly, to the management of resources available to users on desktop computers connected to the network.

2. Description of the Related Art

A. Traditional Data Centers and Networks

As large scale corporate computing has evolved, on-line systems and applications have been made increasingly available to virtually all segments of commerce through the widespread proliferation of desktop computers linked via telecommunication networks to centralized computing centers. Traditionally, these networks have been centrally managed as part of the responsibilities of an information technology organizational unit. The complexity of managing strategically vital computing resources; managing significant numbers of critical business applications and their associated data; providing networked desktop computer access to the appropriate systems and applications; and all the while safeguarding the integrity, security, and availability of such assets have been some of the most significant challenges addressed by both the information management and the computer science disciplin s.

Network systems management software which traditionally has run on centralized computing centers typically is involved with tasks such as, library and configuration management, version control, resource security, network control, data and storage management, job scheduling, session management, resource monitoring and reporting. Much of the earlier network systems management software was developed to operate in highly structured environments readily susceptible to centralized control. In the past, users at desktop computers on the network often had relatively little flexibility in configuring their desktops to meet their specific needs.

Recently, the revolutionary strides in hardware and software technology and in the human-to-computer interface have made an enormous range of different application programs available for desktop use on personal computers. Moreover, desktop computers have become so easy to operate that users have come to rely upon them for a broad array of specialized tasks such as word processing, spread sheet analysis and personal information management. The proliferation of application programs for desktop computers and the wide usage of computers in vast networks continues to make the job of managing networks ever more challenging.

Referring to the illustrative drawing of Figure 1, there is shown an illustrative block diagram of a typical data center which manages

application software and network activities for a network which comprises a multiplicity of end-user terminals such as desktop personal computers. A typical data center includes a central computer, often a powerful mainframe computer, and one or more administrative computer terminals. Ordinarily, programs such as security management programs, configuration change management programs, library management programs and presentation management programs run on the central computer.

B. Migration of Data Center Functions to the Desktop

The development of more powerful desktop personal computers has led to the migration of many functions to the desktop computer. In the past, these functions were managed by a central computer located in the data center. This migration of functions to the desktop has been spurred by the advent of the graphical user interfaces which represent a computing environment in terms of simple pictures in which applications and other resources are depicted as icons to be manipulated. A GUI can permit a user to interact with images on a computer screen in much the same way that he would interact with a real object such as a telephone, a file folder, or other objects that often can be found on a real desk. The resulting easily understood and intuitive environment provides the foundation upon which innumerable new robust and powerful application programs have been built.

As a result, desktop computer users have come to demand from their desktop computers much of the computing functionality that once was reserved for centralized computers and computer centers. In addition, many existing centralized computer applications, and much data critical to an organization, often must be accessible to newly emerging client/server applications. For example, a desktop application for use in sales analysis may require access to product ordering and availability information that is managed by a central computer. Alternatively, for example, a desktop spread sheet application may need data stored in a central database.

In short, a central computer often is called upon to serve as a central repository of services and data that are to be available to different departments within an organization. Different departments often have different needs, and the challenge of centralized management increases as individual users within departments are able to specify their individual needs with greater particularity. Thus, the managers of networks of distributed desktop computers increasingly are being called upon to support a wide range of end-user involvement with the desktop, most notably the productivity enhancements of personalized desktop computing.

C. Content/Distribution Management Issues

The issues in enterprise desktop management include the configuration of networked desktop computers, distribution of resources and ongoing maintenance. Once a desktop computer has been configured with the proper resources, those resources must be properly maintained. One of the challenges confronting an enterprise management system is that there may be a large variety of software resources available on the network and a large number of desktops. The question of, "who gets what?" can be a difficult one. The complexity of this question is increased by the need to specify not only which application programs should be available at which desktops but also by issues such as which version of an application to use, security, policy based limitations and compatibility.

The inability to dynamically reconfigure any or all desktop computers in a network on an as-required basis can be a significant systems management shortcoming that can slow enterprise responsiveness to changing technical and business requirements. For example, in the banking industry, computer software based training information may have to be distributed to hundreds of branch offices and may have to be kept current with changes in bank lending policies and banking regulations. In such a banking scenario, for example, the efficient distribution and updating of the software is critical. Failure to distribute new lending policies or new banking

regulations to all branches could have serious harmful business consequences.

Referring to the illustrative drawing of Figure 2, there is shown a typical enterprise configuration management structure. It includes multiple administrator computer terminals and multiple desktop computers, all of which are connected to a central computer. The administrator terminals allow network administrative personnel to manage the desktops. The central computer provides a centralized locale for the distribution of applications and data to the desktops. The management of such resources can entail maintaining lists of resources available to the desktop computers; or the maintenance of rules for the use of resources on the desktop computers.

Resource availability, for example, could be based on policies, such as policies governing which users and which departments can have access to certain application functions and what business or technical data are to be treated as confidential. The set of resources available to a given desktop, for example, could also depend upon technical issues such as, which operating system is employed on the desktop computer or how much disk space is available on that desktop. The administration of such configurations may involve a combination of list-based and rules-based techniques.

An important challenge in the administration of an enterprise-wide network of desktop computers is that an extremely large body of information often must be managed when the number of desktops, the policies governing them, and the universe of application resources is large. Enterprise systems management involves making certain that the resources possessed by each desktop match the resources that the desktop should possess based upon policy and technology considerations. One approach to managing distributed desktop computer resources is to store in the computer lists of the different resources available on different desktop computers.

For example, computing applications are a type of resource that can be distributed among desktops. An initial deployment of a distributed computing application can require a determination by a policy administrator of which users are authorized to utilize the application, followed by a determination by a technical administrator of the particular version of the application software appropriate for that user's already installed equipment and software environment in order to produce a list of components, or "resources" which must be transported to the user computer over a network connection. Subsequent changes to the deployed application can require an additional process of determining currently authorized users; determining the changed/added/deleted resource(s); and transporting the needed resource changes to the appropriate user(s).

The typical approach to managing these changes is largely a clerical one, requiring a technical administrator to manually create lists of components that need to be changed for the "last known" hardware/software environments of the "last known" list of authorized users. The use of previously developed authorization lists and user environment status creates the possibility of introducing errors into the change process because one or more user authorizations has been granted or revoked since the list was originally created, or because user hardware/software environments themselves changed since their status was last reported. The probability of error is directly related to the accuracy and timeliness of the authorization lists and status information used. Because of the practical difficulties involved in keeping authorization lists and environment status up to date without automated tools and in matching them against components to produce distribution lists without integrating software, the typical change process usually requires a high degree of standardization and a low rate of change in authorizations, environments, and application components.

The illustrative drawing of Figure 3 shows one method for managing and maintaining such stored lists. A list designated as the "Already Have" list is created and saved for each desktop. The AH list lists resources that a desktop already has. Such resources may have been transported to the desktop

during an initial deployment or a previous configuration activity. A list designated as the "Should Have" list can be generated by administrative personnel. The SH list lists resources that a desktop should have. This list itemizes the resources that should be available to the desktop for all authorized activities for that desktop. The AH lists are stored at either the central computer or at the local desktops. Ordinarily, the SH lists are stored centrally or at the administrative computer that created them. However, the storage location of the AH and SH lists is not critical.

The list designated as the "Need" list is generated by comparing the AH list with the SH list. Items contained in the SH list that are absent from the AH list are included in the "Need" list. For example, the "Need" list indicates an addition of a new item designated as "ggg" which is listed in the SH list but which is not listed in the AH list. The "Need" list also sets forth items which are present in the AH list, but which have been updated in the SH list. These updated items in the "Need" list are to replace corresponding items in the AH list. For example, the item listed as "aaa" in the SH list was last updated on date, 05/01/92, at time, 12:13:45 o'clock. The corresponding item indicated as "aaa" in the AH list was last updated on date, 03/01/92, at time, 12:13:45 o'clock. Thus, the "Need" list includes the item "aaa" as updated on 05/01/92 at 12:13:45 o'clock which will supplant the item "aaa"

item last updated 03/01/92. Moreover, the "Need" list identifies items that are present in the AH list but absent from the SH list, and which, therefore, must be deleted from the desktop. The item "fff" is listed in the "Need" list as requiring deletion from the AH list.

The items listed in the AH list, such as for example, "ddd" can represent resource names or pointers to resources or data sets. The dates and times adjacent to such items indicate the currency level of the resource. For example, the item identified as "ddd" in the SH list was last updated on the day of 05/14/92 at 13:11:45 O'clock. The "Need" list identifies resource deletions, additions and updates necessary to configure a desktop computer in accordance with administrator requirements designated in the SH list. Thus, one earlier process of configuring a desktop computer involved determining what resources the desktop already has; determining what resources the desktop should have; and based on a comparison of the AH and SH resources, determining what resources the desktop needs to add, delete or update. Therefore, once a "Need" list is generated, it is used to manage the process of deleting resources from the desktop or of adding resources to the desktop or updating resources on the desktop.

For example, earlier processes of deleting, adding or updating resources for example, can involve transporting new or updated versions of

programs or data over the network and deleting no longer needed versions of programs or data from the desktop. The types of resources that can be deleted, added or updated using such earlier processes can include virtually any information that can be transported over the network such as programs, data, icons and batch files.

One problem with the type of lists described with reference to Figure 3 is that the addition, deletion and updating of desktop resources can involve the searching/comparing of entire SH lists to entire AH lists for a large number of desktops which can involve large network transport overhead. The complexity of the search/comparison task increases both with the number of desktops and with the number of resources managed. Referring to the illustrative drawings of Figure 4, there is shown a proliferation of AH lists and of corresponding SH lists. Each AH list and each corresponding SH list relates to a different desktop. The process explained with reference to Figure 3 must be performed for every desktop in the network.

An earlier approach to simplifying the management of large numbers of desktops and resources is to group desktop computers together to reduce the number of lists. Instead of maintaining a different AH and SH list for each desktop, different AH and SH lists are maintained for groups of desktops. As a result, the total number of AH and SH lists is reduced. However, such grouping can

require standardization of the resources available to groupings of desktops, which reduces the user's ability to customize their individual desktops. Another similar alternative could use a rules-based approach to the grouping of desktops. For example, desktops could be divided into groups, and rules could be imposed which assign one set of resources to one group and another set of resources to another group. The result, however, is the same: a reduction in the user's ability to customize their individual desktops.

The grouping of resources permits use of structured SH lists which are SH lists made up of sub-lists. The use of structured SH lists can reduce storage requirements for SH lists by obviating the need to store a separate set of lists for each desktop. The use of structured SH lists can also simplify the implementation of changes, since changes to structured lists can have effect across numerous desktops on the network. Referring to the illustrative drawings of Figure 5, there is shown an SH list which is a composite of two sub-lists. As illustrated in Figure 6, sub-lists can be shared among SH lists.

Thus, items can be grouped into sub-lists by an administrator based on an analysis of resource sharing among desktops. When a desktop request for a configuration activity prompts a comparison of a SH list for a desktop with the AH list for that desktop, then the SH list can be generated from the

master list and the sub-lists. One difficulty with automating the SH list generation process has been that different desktops can have different hardware and software platforms. Thus, the SH list generation process frequently involved a clerical task in which administrative personnel determined which resources a desktop should have.

In order to ease the administrative burden of generating SH lists, desktops often are grouped together and share common SH lists. Once the SH list has been generated for a desktop or a group of desktops, then the SH list can be compared to the AH lists of the individual desktops to generate "Need" lists. However, as explained above, grouping of desktops involves a cost in terms of granularity of the available resources: desktop users may be relegated to the nearest fit sub-list or the least common denominator of sub-lists due to an inability to fully articulate and manage the specific desktop user configuration requirements for individual desktops.

Another problem with the use of structured SH lists is that the desktop platform may change, and resources identified in a SH sub-list may be inappropriate in the changed desktop platform. For example, the type of desktop printer may be changed; the monitor type may change; the operating system or the network operating system may be changed to an updated version. Consequently, a desktop, may require different SH sub-lists which list resource

items appropriate for the changed desktop platform. Rules-based lists can be more dynamic in that they can accommodate such changes automatically. For example, rules like "if printer type A then use SH sub-list A, and if printer type B, then use sub-List B," could be applied. However, such rules can require additional programming effort by network administrative personnel to anticipate the multitude of potential desktop platform changes.

Therefore, there has been a need for an improved and automated configuration process that permits dynamic reconfiguration of a desktop based upon policy and desktop platform changes.

There also has been a need to optimize the process of comparing SH lists with AH lists to determine what resources must be added, deleted or updated on a desktop computer. This need is particularly acute in networks having a large number of desktops and a large number of variable resources. In such large networks, a one-to-one approach of matching an AH list against a generated SH list for a large number of desktops and a large number of resources can be storage, processing and network transport intensive. For example, if there are 5,000 desktops and 1,000 possible resources, then there could be five million records in the AH and SH lists. If there are 100 characters per record then a Gigabyte of data might have to be stored. The present invention also meets the need to improve the comparison process.

There also has been a need for a method of network administration which automatically makes resources available to a desktop in response to predictable changes in the desktop platform. The present invention also meets this need.

SUMMARY OF THE INVENTION

In one aspect of the invention, dynamic linkage substitution is used to facilitate the resolution of a SH list so as to meet the current predictable needs of a desktop environment. In another aspect of the invention, an LUDT/LSDT comparison process is used to avoid unnecessary comparisons of AH list items with SH list items. In still another aspect, both dynamic linkage substitution and an LUDT/LSDT process are employed.

The present invention automates and integrates the management of changes in a distributed computing environment by automatic construction of required/needed component lists for each distributed user based on a real-time interrogation of the user's environmental status together with the user's current authorization as derived in real-time from access rules and/or lists maintained by policy administrators. Integral to this automation is the process of "differencing" or difference calculation between already distributed components and the components required by changes to the user's environment, authorization, or the application itself. Since the process itself "remembers" users and already distributed components and automatically

calculates needed component lists based on current user authorization and environment status, the administrative activities are simplified while accuracy and precision are improved. Because this automated process is equally applicable and effective for the initial deployment of an application it can render the change management process virtually indistinguishable from the initial configuration process.

These and other purposes and advantages of the present invention will become more apparent to those skilled in the art from the following detailed description in conjunction with the appended drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram illustrating the prior art relationship between a datacenter which includes, administrator terminals and a mainframe computer, and end-users with desktop computers;

Figure 2 is a generalized drawing of a typical enterprise management structure which includes a central computer administered through multiple administrator terminals and which further includes a multitude of desktops.

Figure 3 is an illustrative drawing showing that a comparison of an Already Have list and a Should Have list results in a Need list which itemizes discrepancies between the AH and SH lists;

Figure 4 is an illustrative drawing showing the potential for a proliferation of AH lists and SH lists for a network that includes a multitude of desktops;

Figure 5 is an illustrative drawing showing the use of a structured SH list which comprises two sub-lists; the SH list is compared with an AH list to arrive at a Need list;

Figure 6 is an illustrative drawing which shows the sharing of sub-lists among different structured SH lists;

Figure 7 is an illustrative drawing which illustrates the resolution of a dynamic linkage as part of the process of generating an SH list from sub-lists in accordance with the present invention;

Figure 8 shows an illustrative schema which includes a dynamic linkage in accordance with the present invention;

Figure 9 illustrates a Last Updated Date/Time (LUDT) field in an SH list and a Last Synchronized Date/Time (LSDT) field in an AH list in accordance with the invention;

Figure 10 illustrates the use of the LUDT/LSDT process in accordance with the present invention;

Figure 11 illustrates the use of an Object Identity Checksum (OIDC) and a Last Synchronized Checksum (LSC) to detect dynamic linkage substitution in a Should Have list in accordance with the present invention; and

Figure 12 illustrates the process of fractional change detection by parsing a SH list schema using both an LUDT/LSDT process and an OIDC/LSC process in accordance with the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention comprises a novel method and apparatus for enterprise system management of the resources of distributed desktop computers. The following description is presented to enable any person skilled in the art to make and use the invention, and is provided in the context of particular applications and their requirements. Various modifications to the preferred embodiment will be readily apparent to those skilled in the art, and the generic principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the invention. Thus, the present invention is not intended to be limited to the embodiment shown, but is to be accorded the widest scope consistent with the principles and features disclosed herein.

Adaptive List Generation Using Dynamic Substitution

Referring to the illustrative drawing of Figure 7 there is shown a process of adaptive list generation using structured lists in accordance with the present invention. A process of dynamic linkage substitution allows for dynamic linkage of sub-lists which can facilitate automatic adaptation of a SH list to predictable user initiated changes in the desktop environment. That is, if the user changes the desktop environment, the SH list adapts accordingly.

In a dynamic linkage process in accordance with the present invention, a branch interrogation such as "Size?" in Figure 7, results in a substitution of a value, "Small" or a value, "Large" based upon a desktop resource variable. The substituted value, Small or Large, is determined based upon the desktop platform at the time of SH list generation. Thus, the generated SH list reflects up-to-date desktop resource needs.

Referring to the illustrative drawing of Figure 8, there is shown a representative schema for a desktop SH list in accordance with the present invention. The overall SH list comprises sub-lists A, B and C. Sub-list A includes sub-lists A1 and A2. Sub-list B includes sub-lists B1, B2 and B3. Sub-list C includes sub-lists C1 and C2. Sub-list B2 includes sub-lists B21, B22 and B23. The linkage between sub-list B2 and sub-lists B22 and B23 is a dynamic linkage. At the time of the SH list

resolution, only one of the sub-lists B22 and B23 will be substituted into the SH list schema. The steps whereby the schema is parsed to resolve the SH list are described below.

The use of dynamic linkage substitution to accommodate desktop platform changes will be better appreciated through the following illustrative example. One desktop resource variable could be the operating system which could be OS/2™ or Windows for example. A change in the desktop operating system variable can change a sub-list of resources that a desktop should have. During a configuration process, the desktop is interrogated as to all variables such as operating system, microprocessor type, disk space, memory size, interrupts and installed application software. The purpose of the interrogation process is to assign names for each such variable. The operating system variable can be represented by the variable "Size" in Figure 7, for example, and possible values for the Size variable could be "Small" for Windows and "Large" for OS/2.

Thus, a naming convention is used to determine values for resource variables. Interrogation Code selects variable names that correspond to the resource values in use on the desktop. At the time of SH list resolution, the variable names can be transported to the computer, via standard network connections using standard network protocols, where the SH list resolution process takes place. the SH list resolution process ordinarily occurs at the

central computer. However, it could be performed at the desktop instead. List resolution is part of the configuration process which can be triggered by any number of events such as a desktop timer, the "boot procedures," or by the user selecting an appropriate icon. During SH list resolution, a sub-list is linked into the SH list based on the name assigned for the variable during the interrogation process. This process occurs for each desktop variable resource. Thus, the desktop SH list is responsive to predictable desktop platform changes.

Thus, the dynamic linkage process uses naming conventions. The sub-lists associated with a dynamic linkage correspond to different variables. The name which is assigned to the variable during the interrogation process determines how the dynamic linkage will be resolved during SH list generation.

An administrator establishes these different names. The Interrogator Code associates the names to resource variables. The naming conventions permit different sub-lists, for example, to be associated with different desktop variables. Continuing with the above example in which "Small" is associated with a Windows operating system and "Large" is associated with an OS/2 operating system, Windows requires one set of resources which correspond to the items in the sub-list named, Small; and OS/2 requires another set of resources which correspond to the items in the sub-list named, Large. The different sub-lists, for example, could

list different applications which use different graphical interface modules for use with Windows on the one hand and with OS/2 on the other hand.

Since it is predictable that a desktop environment operating system variable may charge from one variable type to another variable type, a dynamic linkage can be built into the structured SH list. When the SH list is generated the linkage will be generated in favor of a sub-list that reflects the resource needs of the desktop operating system type.

An advantage of adaptive list generation using dynamic linkage substitution is that it can reduce the need for administrator intervention when the desktop environment is changed. The use of naming conventions in accordance with the present invention permits the resolution of an SH list which reflects the particular needs of a desktop environment without resort to rule based programming. The novel use of dynamic linkages, therefore, advantageously permits desktop users to customize their desktops to fit their particular requirements without imposing an undue management burden upon the network administrators. The use of dynamic linkages affords the automatic connection of sub-lists appropriate to the desktop environment based upon the value/name assigned for a particular variable in a desktop environment.

It will be appreciated that the configuration process in accordance with the present invention permits the automation of system administration by separating policy decisions from technology packaging decisions. The use of variables and dynamic linkages in the technology packaging process permits technology packaging requirements to automatically conform to both policy changes and to desktop platform changes.

Comparison Avoidance Using Last Synchronization Date/Time

Referring to the illustrative drawings of Figure 9, there is shown the use of a Last Updated Date/Time field and the Last Synchronized Date/Time field which can reduce the network transport overhead associated with the comparison of SH list items. The LUDT for an item in an SH list represents the last date/time on which the item was updated or added to the SH list. The LSDT represents the last date/time on which the SH list and the AH list were synchronized. A one-to-one comparison of all SH list items to all AH list items can be avoided when there are no changes to previously configured resources. By comparing the LUDT field for each resource entry in a generated SH list with the LSDT for a desktop AH list, a determination can be made as to whether or not a comparison of the items SH and AH lists is necessary. If all of the LUDTs in the SH list are earlier than the LSDT for the desktop AH list, then no previously defined resources in the SH list have

changed since the last synchronization of the SH and AH lists for the desktop and no lists need to be compared for changes.

Referring to the illustrative drawing of Figure 10, another approach in accordance with the invention is to maintain an overall LUDT for the entire SH list. This approach requires maintenance of overall LUDTs for each sub-list within the SH list. If the overall LUDTs for each of the SH list sub-lists for a particular desktop are less than the desktop LSDT then the contents of the generated SH list resource list could not have been altered by addition, deletion or update since the last synchronization of the SH and AH lists for the desktop. This approach shall be referred to as Enhanced LUDT. The LUDT or ELUDT processes can be used during the SH list generation process in order to bypass the one-to-one comparison of all AH list contents with all SH list contents when no additions, deletions or updates have been introduced to the SH list since the date/time of the last synchronization.

It will be appreciated that LUDT is useful for "flat" lists, while ELUDT is useful for structured lists. The LUDT can identify changed resources. The ELUDT can identify the addition or deletion of entire sub-lists.

The drawing of Figure 10 shows the use of the LUDT process in connection with SH list. The

desktop LSDT for the desktop AH list is compared with the overall SH list LUDT and with the LUDTs for each of the two SH sub-lists to determine whether any one-to-one sub-list comparison is necessary. In this case, since the LSDT is later than each of the three LUDTs associated with the SH list, no further comparison is necessary.

The ELUDT process works not only for sub-lists, but also for equivalent intermediate connecting structures such as objects. For example, if objects are maintained instead of lists then current ELUDT information is maintained for the objects. Keeping track of ELUDTs for a master list and for all sub-lists is tantamount to inheritance of ELUDT information within object classes.

It will be appreciated that the ELUDT information is checked as a prelude to the actual process of comparing the actual contents of SH and AH lists. Actual comparisons are required only when SH list or sub-list have been changed since the list synchronization of the desktop AH and SH lists. Moreover, as explained below, a comparison process in accordance with the present invention seeks to compare as little as necessary to bring the AH list up to date with the SH list.

Fractional Change Detection

As explained above, dynamic linkage substitution occurs at the time of SH list generation. A potential problem is that the desktop environment may have changed after the last synchronization of the AH and SH lists for a desktop. Specifically, the LUDT of each SH list and each SH sub-list used in the most recent synchronization might be older than the LSDT, but the desktop platform nevertheless may have changed since the most recent synchronization. That is, after the last synchronization a desktop variable may have changed. This change in variable, such as a change in monitor types for example, will not ordinarily be detectable through the LUDT/LSDT process alone. Moreover, the change in desktop variables may change the resources that a desktop should have.

A lookaside table of substitution values used during the last SH list generation process can be stored so as to identify required SH list changes due to changes in the desktop variables.

Another process in accordance with the invention for identifying desktop changes that may require a change in the resources that a desktop should have involves the use of dynamic linkage substitution and a checksum process. A unique numerical identifier which shall be called an object identifier is assigned to each resource and to each

sub-list and to any other connecting elements or sub-sub-lists used to generate the list.

During the generation of an SH list, the OID of each resource or sub-list included in the SH list is processed using one of two alternative checksum methods. An overall OID checksum for the SH list is calculated. If the overall OIDC is equal to a Last Synchronized Checksum for the user desktop, then the overall structure of the generated SH list has not changed. The LUDT process is used to detect changes in the contents of any particular list or sub-list. The OID process detects linkage changes. The LUDT process detects changes in the contents of individual resource lists or sub-lists. The OID checksum process, together with the LUDT process allows the one-to-one AH list to SH list comparison to be bypassed in cases where there have been no changes, even where dynamic linkage substitution or other dynamic or generation techniques have been employed.

Referring to the illustrative drawing of Figure 11, there is shown in an example of change detection using an OID checksum to detect dynamic linkage substitution. Since the desktop LSDT is older than or equal to all of the LUDTs in the SH list, no previously defined SH list resources have changed, at least not resources which cannot be subject to dynamic substitution. In addition, the overall OIDC of the SH list is compared with the LSC of the desktop AH list. If the overall OIDC is not equal

to the LSC, then there has been a change in the composition of the sublists. The overall most recently generated OIDC becomes the new LSC, and further comparison is required to detect the detailed changes. Figure 12 which shows an example of such further comparisons as part of a process of fractional change detection in accordance with the present invention.

The process of fractional change detection detects both changes in linkages and changes within resource sublists. In particular, for example, a checksum process can be used to determine whether the linkage substitution of B22 or B23 in Figure 9 is different from the linkage substitution the last time the AH and SH lists were synchronized. The LUDT process is used to determine whether an existing resource entry within any of the sub-lists has changed. For example, a sub-list may have resource "qqq". The parameters within resource "qqq" may have changed, however. This change in resource "qqq" parameters is identified by date changes. Only sub-lists and resources for which changes have occurred require comparison as explained in relation to Figure 12.

Referring to Figure 12, there is shown a fractional date change detection process. A latest LUDT is generated for the overall SH list while the SH list is being generated, and an LLUDT is maintained for the AH list. Similarly, for each sub-list, a separate LLUDT is maintained. In order

to avoid further comparison, the LLUDT for a given sub-list will have to be earlier than the LSDT. The fractional change detection process involves separate change detections and comparisons for the different sub-lists. The LLUDT for sub-list A indicates the latest LUDT out of all of the resource entries in sub-list A. The LLUDT for sub-list B, indicates the latest LUDT out of all of the entries in sub-list B. Similarly, for the AH list, an LLUDT is maintained for sub-list A, and an LLUDT is maintained for sub-list B. Thus, LUDT and LLUDT values are maintained both for the SH list and for the AH list.

An SOIDC is generated while the SH list is generated. A desktop LSC is saved for the AH list each time the AH list and the SH list are synchronized during the configuration process. The LLUDT/SOIDC information for the SH list is compared with the LSDT/LSC for the overall AH list. If the LLUDT matches the LSDT, and the SOIDC matches the LSC, then no further comparison of linkages or resource entries in the AH and SH lists is necessary. If, they are not equal, however, then the change detection/differencing process proceeds further.

In particular, the comparison of the SOIDC and the LSC for sub-lists A of the SH list and the AH list shows that for these sub-lists, the SOIDCs and LSC's match. Thus, there has been no change in the linkages or the resource information in sub-list A

for the SH list or in sub-list A for the AH list. For sub-list B, however, the SOIDCs are different. This indicates a linkage change. Moreover, the LUDTs are different as well. This indicates an update in a resource entry in sub-list B. Thus, the contents of sub-list B of the SH list no longer match the contents of sub-list B of the AH list. Further comparison of the AH sub-list B and the SH sub-list B are required. This further comparison can involve network transport of the SH sub-list to the desktop for comparison with a desktop AH sub-list. In particular, referring to the specific contents of sub-list B of the SH list, the file identified as "eee" having the OID 005 is different from the file identified as "fff" which has an OID of 009.

It will be appreciated that the above description is an example of fractional change detection because comparisons are performed on fractional parts of the schema rather than on the entire schema. No changes between the SH sub-list A and the AH sub-list A were detected. Hence, no further comparison of the SH sub-list A and the AH sub-list A were undertaken. However, changes between the SH sub-list B and the AH sub-list B were detected. Hence, further comparison of the AH sub-list A and the SH sub-list B were performed.

Various modifications to the preferred embodiment can be made without departing from the spirit and scope of the invention. Thus, the foregoing description is not intended to limit the invention which is described in the appended claims in which:

WHAT IS CLAIMED IS:

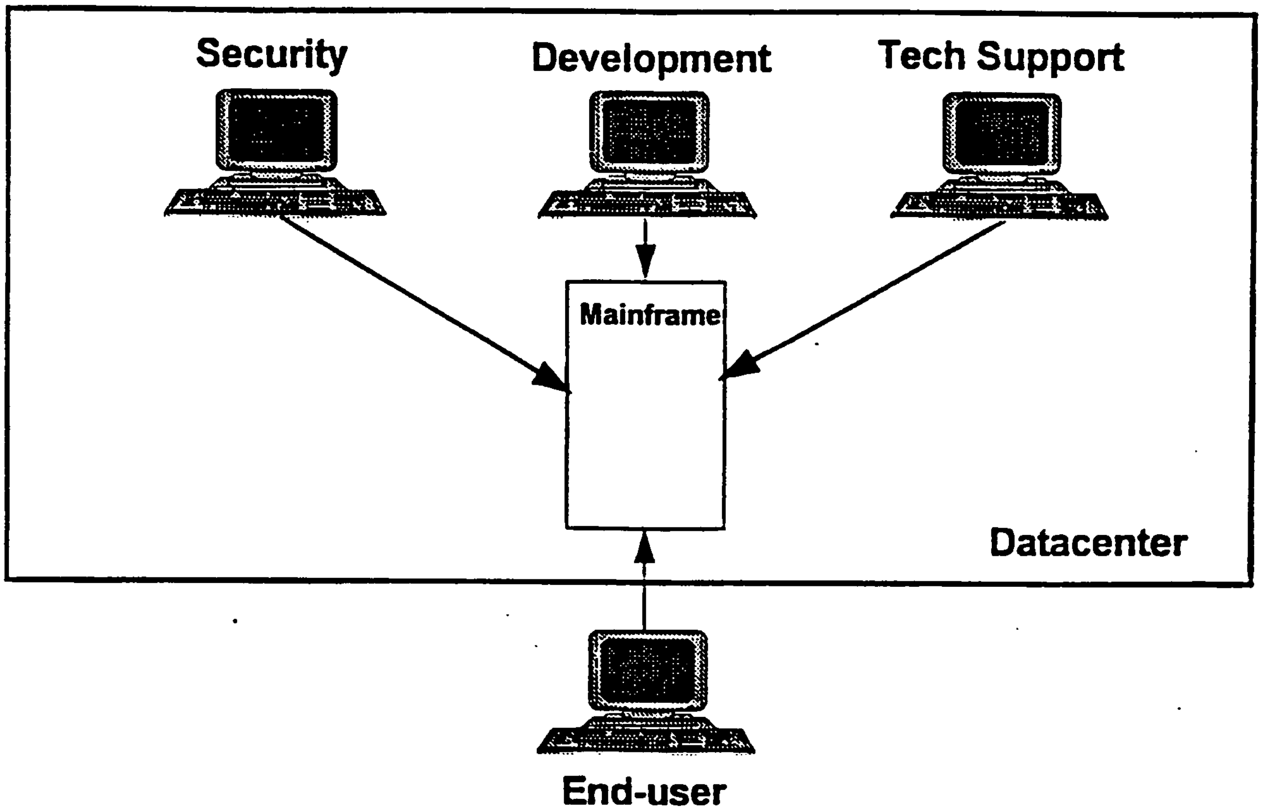
1. A process of enterprise system management for use with an enterprise-wide network which includes at least one centralized computer and a plurality of desktop computers, the method comprising the steps of:

storing an Already Have list for each desktop;

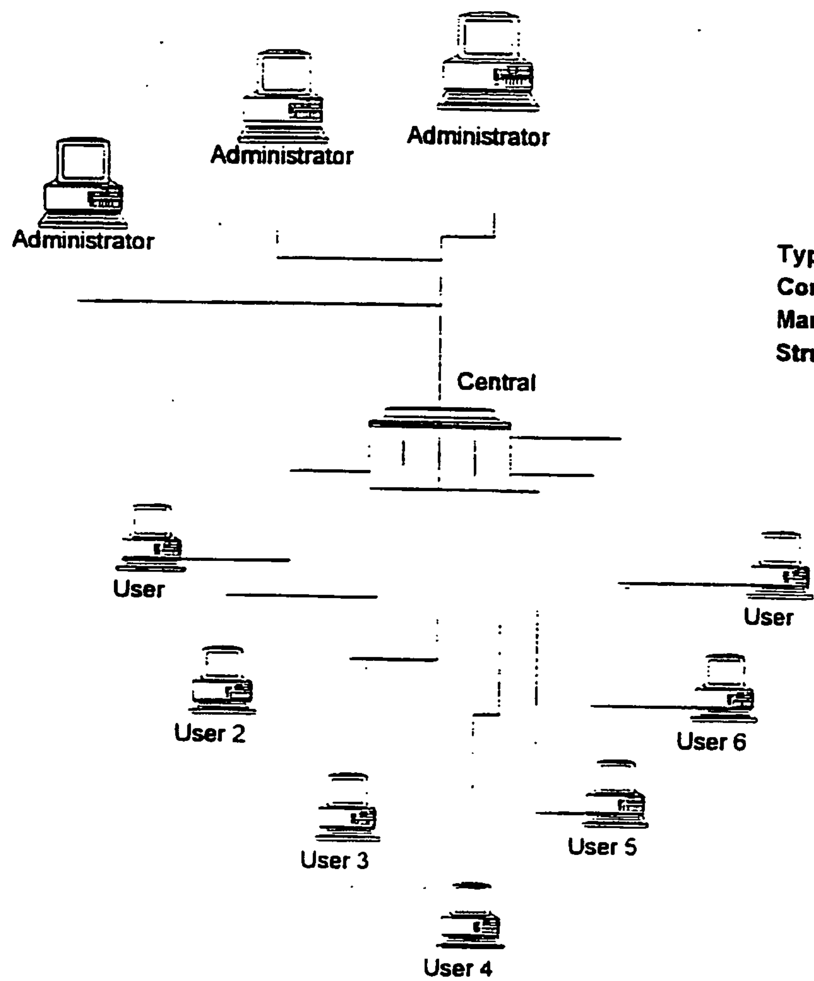
storing a plurality of Should Have sub-lists;

generating a respective Should Have list from the stored sub-lists for a respective desktop computer during configuration of the desktop computer; and

wherein the schema of the generated Should Have list includes at least one dynamic linkage which encompasses more than on Should Have sub-lists.



PRIOR ART - FIGURE 1



Typical Configuration Management Structure

FIGURE 2

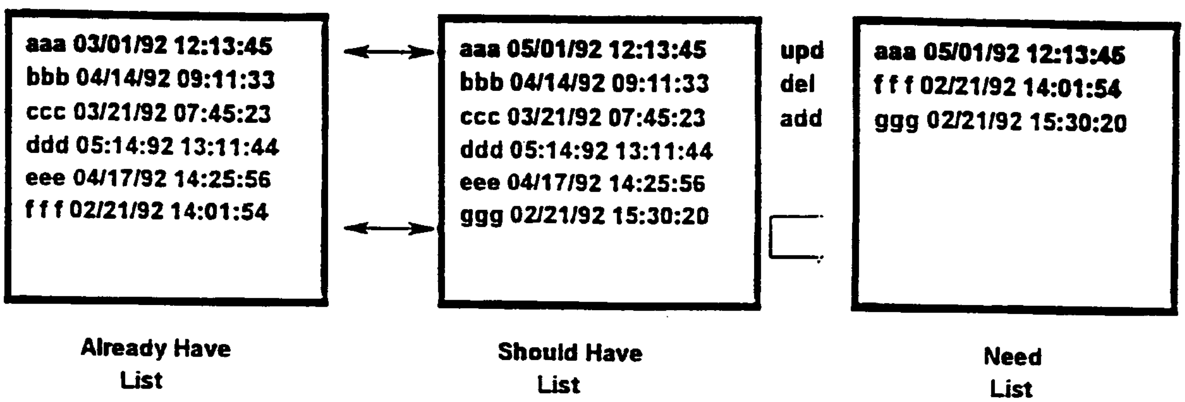

aaa 03/01/92 12:13:45
bbb 04/14/92 09:11:33
ccc 03/21/92 07:45:23
ddd 05:14:92 13:11:44
eee 04/17/92 14:25:56
f f f 02/21/92 14:01:54
Already Have List
aaa 05/01/92 12:13:45
bbb 04/14/92 09:11:33
ccc 03/21/92 07:45:23
ddd 05:14:92 13:11:44
eee 04/17/92 14:25:56
ggg 02/21/92 15:30:20
Should Have List
upd
del
add
aaa 05/01/92 12:13:45
f f f 02/21/92 14:01:54
ggg 02/21/92 15:30:20
Need List

FIGURE 3

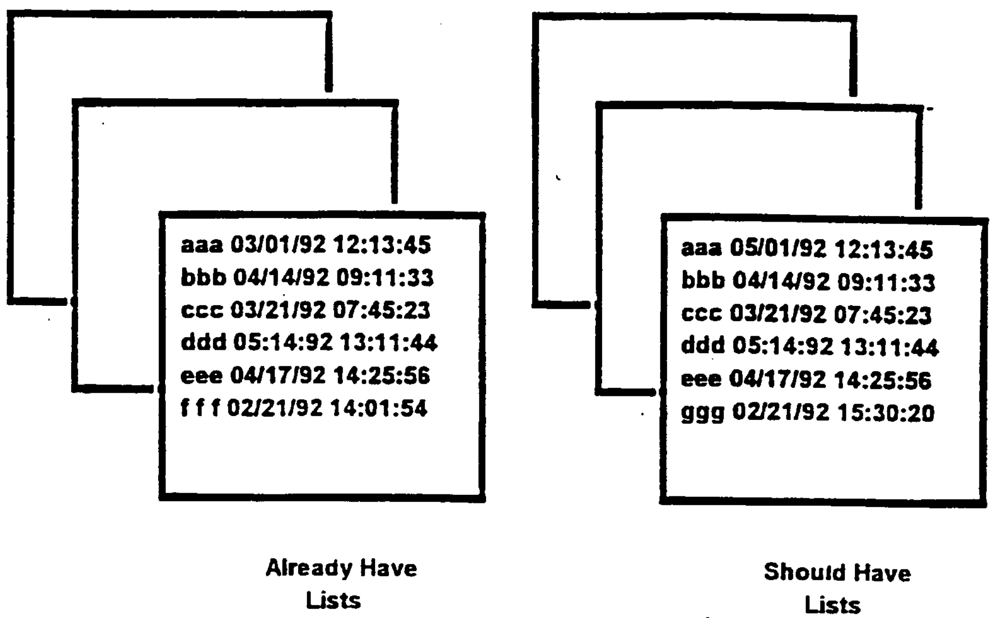

aaa 03/01/92 12:13:45
bbb 04/14/92 09:11:33
ccc 03/21/92 07:45:23
ddd 05:14:92 13:11:44
eee 04/17/92 14:25:56
f f f 02/21/92 14:01:54
Already Have Lists
aaa 05/01/92 12:13:45
bbb 04/14/92 09:11:33
ccc 03/21/92 07:45:23
ddd 05:14:92 13:11:44
eee 04/17/92 14:25:56
ggg 02/21/92 15:30:20
Should Have Lists

FIGURE 4

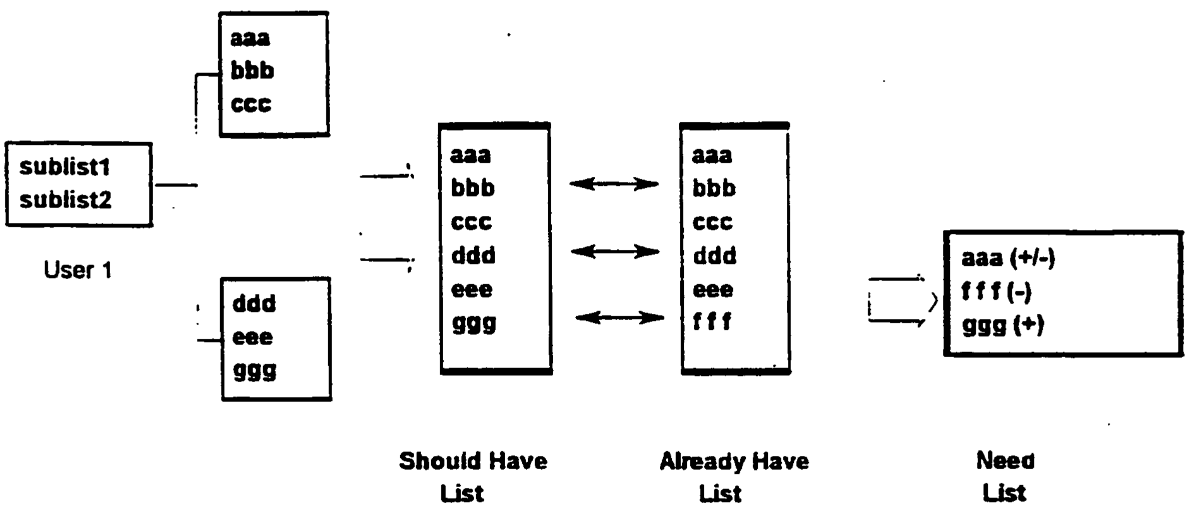
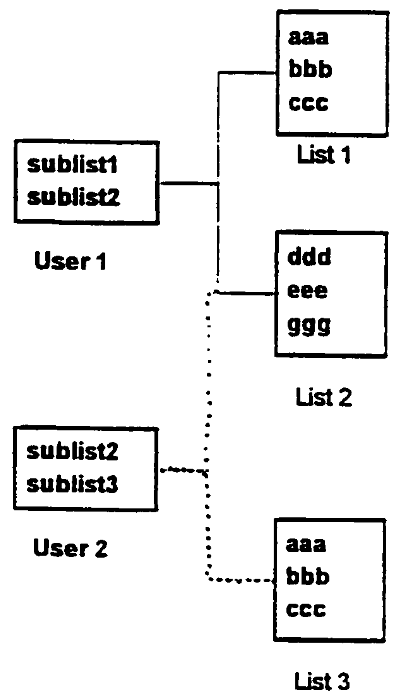
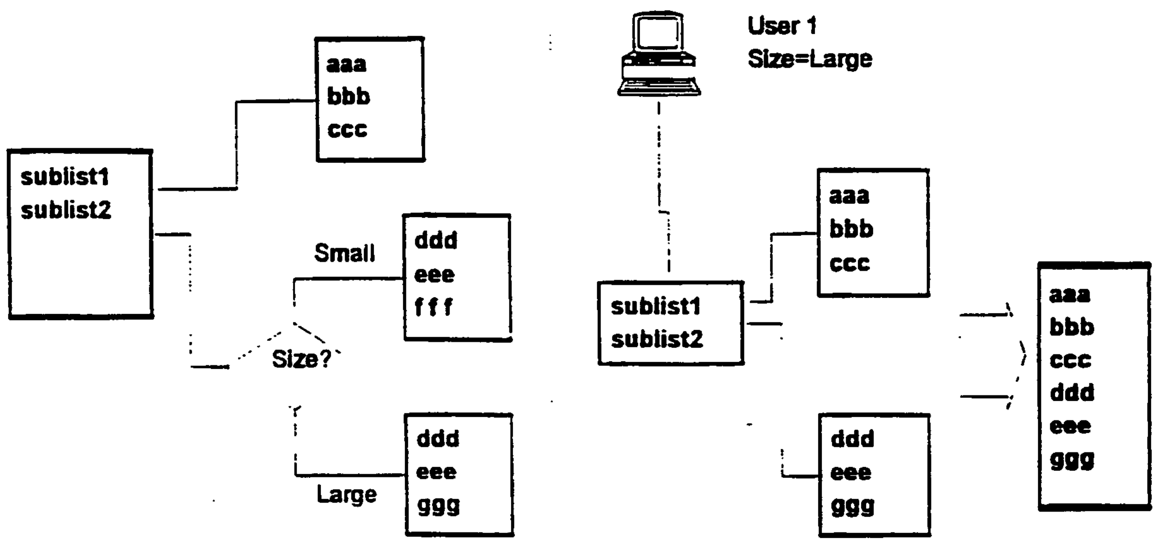

aaa
bbb
ccc
sublist1
sublist2
User 1
ddd
eee
ggg
aaa
bbb
ccc
ddd
eee
ggg
aaa
bbb
ccc
ddd
eee
f f f
aaa (+/-)
f f f (-)
ggg (+)
Should Have List
Already Have List
Need List

FIGURE 5



Sublist Sharing Example

FIGURE 6



Should Have List Configuration

Should Have List Generation
For User 1 with Size=Large

FIGURE 7

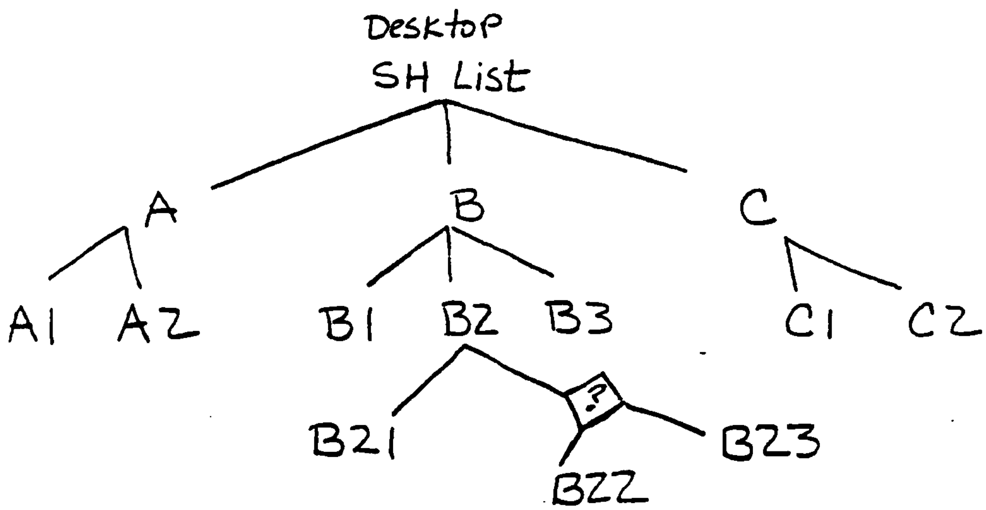


FIGURE 8

Name	LUDT
aaa	05/01/92 12:13:46
bbb	04/14/92 09:11:33
ccc	03/21/92 07:45:23
ddd	05/14/92 13:11:44
eee	04/17/92 14:25:56
ggg	02/21/92 15:30:20

Should Have List

LSDT
05/14/92 13:11:44

LSDT is >= any LUDT in AHL.

Already Have List

FIGURE 9

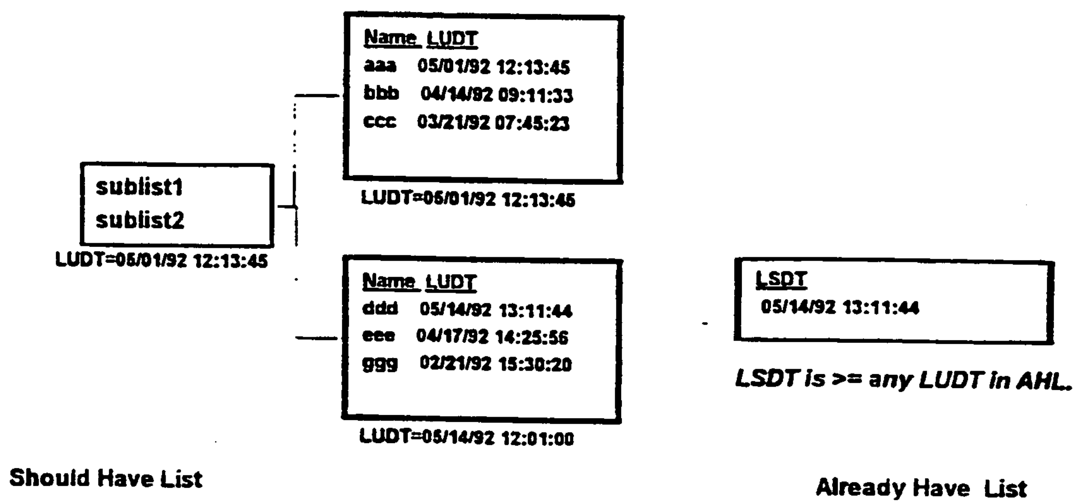

sublist1
sublist2
LUDT=05/01/92 12:13:45
Name LUDT
aaa 05/01/92 12:13:45
bbb 04/14/92 09:11:33
ccc 03/21/92 07:45:23
LUDT=05/01/92 12:13:45
Name LUDT
ddd 05/14/92 13:11:44
eee 04/17/92 14:25:56
ggg 02/21/92 15:30:20
LUDT=05/14/92 12:01:00
Should Have List
LSDT
05/14/92 13:11:44
LSDT is >= any LUDT in AHL.
Already Have List

FIGURE 10

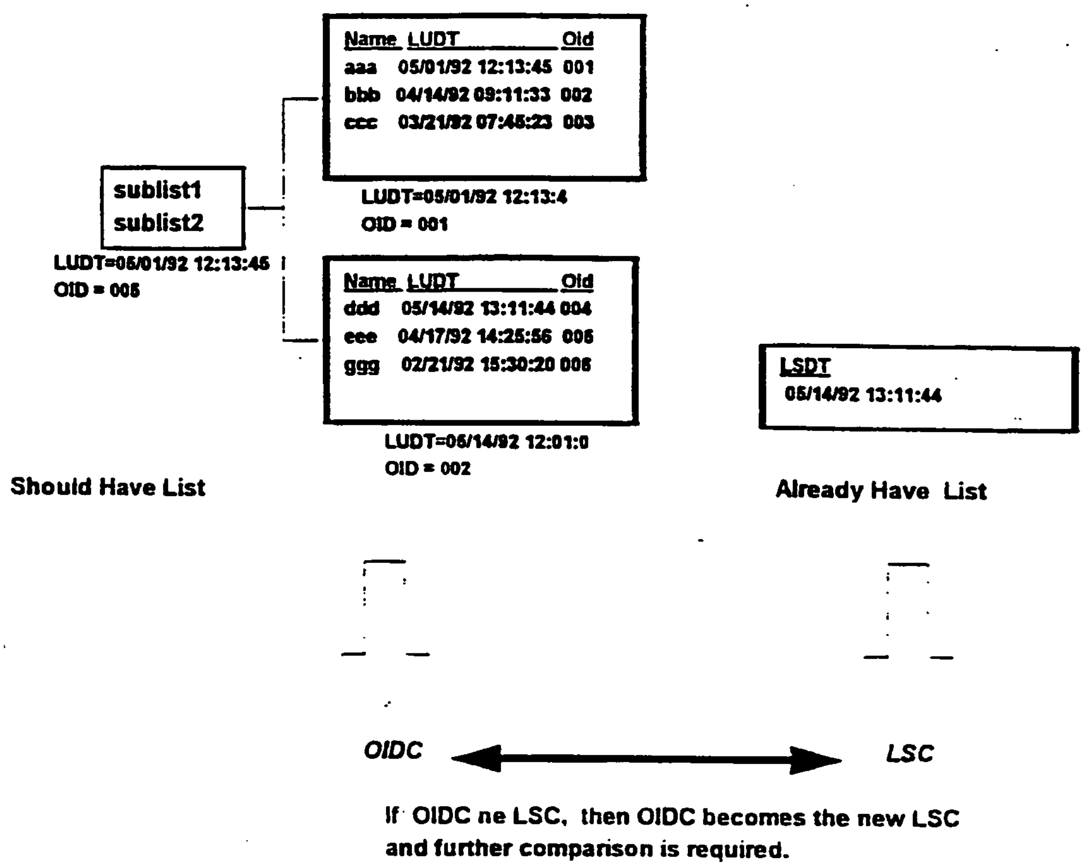

Name LUDT Oid
aaa 05/01/92 12:13:45 001
bbb 04/14/92 09:11:33 002
ccc 03/21/92 07:45:23 003
sublist1
sublist2
LUDT=05/01/92 12:13:4
OID = 001
LUDT=05/01/92 12:13:45
OID = 005
Name LUDT Oid
ddd 05/14/92 13:11:44 004
eee 04/17/92 14:25:56 005
ggg 02/21/92 15:30:20 006
LSDT
05/14/92 13:11:44
LUDT=05/14/92 12:01:0
OID = 002
Should Have List
Already Have List
OIDC
LSC
If OIDC ne LSC, then OIDC becomes the new LSC
and further comparison is required.

FIGURE 11

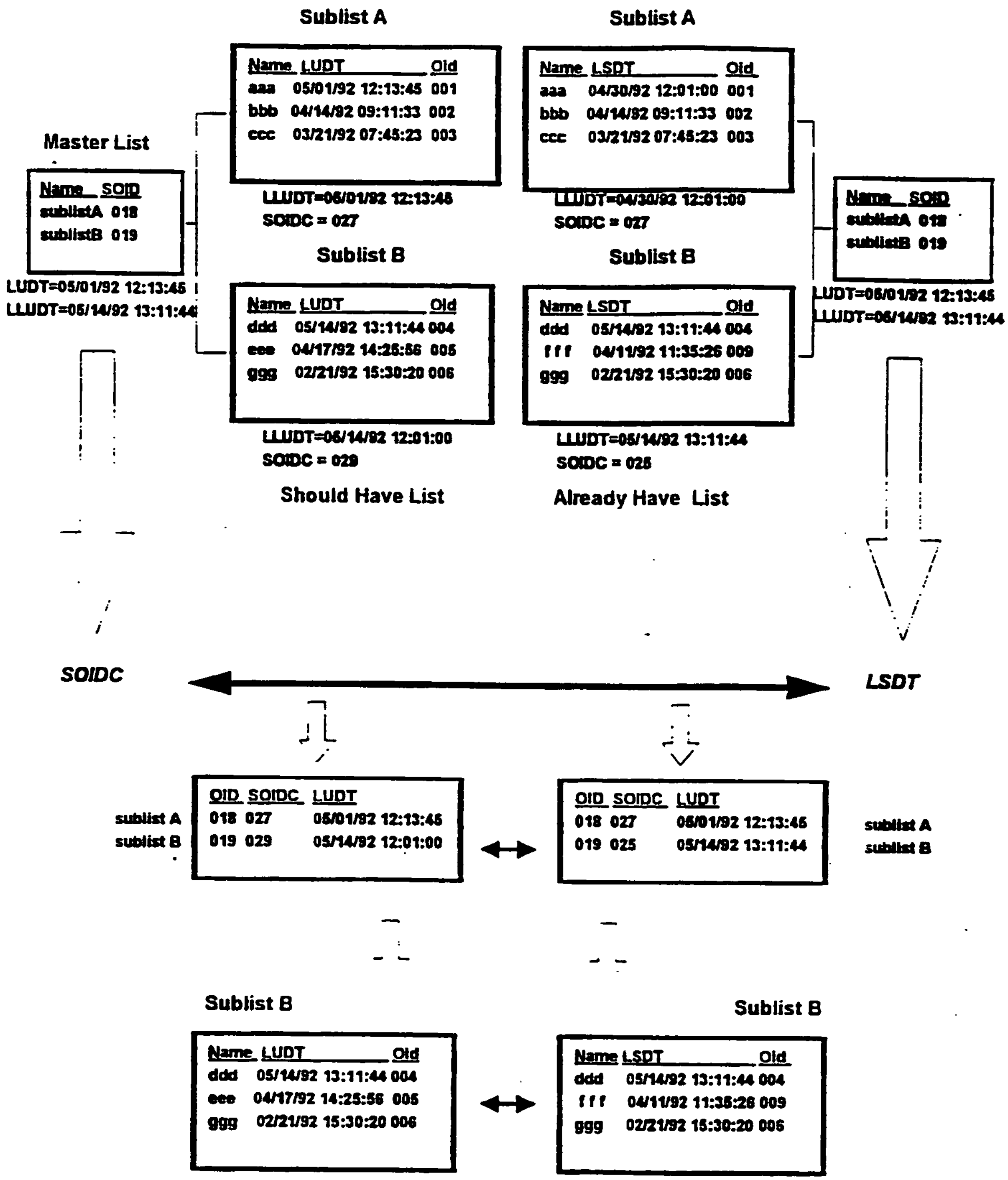

Sublist A
Name LUDT Oid
aaa 05/01/92 12:13:45 001
bbb 04/14/92 09:11:33 002
ccc 03/21/92 07:45:23 003
LLUDT=05/01/92 12:13:45
SOIDC = 027
Sublist A
Name LSDT Oid
aaa 04/30/92 12:01:00 001
bbb 04/14/92 09:11:33 002
ccc 03/21/92 07:45:23 003
LLUDT=04/30/92 12:01:00
SOIDC = 027
Master List
Name SOID
sublistA 018
sublistB 019
LUDT=05/01/92 12:13:45
LLUDT=05/14/92 13:11:44
Name SOID
sublistA 018
sublistB 019
LUDT=05/01/92 12:13:45
LLUDT=05/14/92 13:11:44
Sublist B
Name LUDT Oid
ddd 05/14/92 13:11:44 004
eee 04/17/92 14:25:56 005
ggg 02/21/92 15:30:20 006
LLUDT=05/14/92 12:01:00
SOIDC = 029
Sublist B
Name LSDT Oid
ddd 05/14/92 13:11:44 004
fff 04/11/92 11:35:26 009
ggg 02/21/92 15:30:20 006
LLUDT=05/14/92 13:11:44
SOIDC = 025
Should Have List
Already Have List
SOIDC
LSDT
sublist A
sublist B
OID SOIDC LUDT
018 027 05/01/92 12:13:45
019 029 05/14/92 12:01:00
OID SOIDC LUDT
018 027 05/01/92 12:13:45
019 025 05/14/92 13:11:44
sublist A
sublist B
Sublist B
Name LUDT Oid
ddd 05/14/92 13:11:44 004
eee 04/17/92 14:25:56 005
ggg 02/21/92 15:30:20 006
Sublist B
Name LSDT Oid
ddd 05/14/92 13:11:44 004
fff 04/11/92 11:35:26 009
ggg 02/21/92 15:30:20 006

FIGURE 12